FOCUS

Introduction

Codon optimization is a technique often used by molecular biologists to increase protein expression yields by augmenting the rate translation for a specific gene of interest. Each organism has a codon usage bias where it prefers to use certain codons to encode for a specific amino acid. Current methods of codon optimization work by substituting less frequent codons for a particular amino acid to a more commonly used codon.

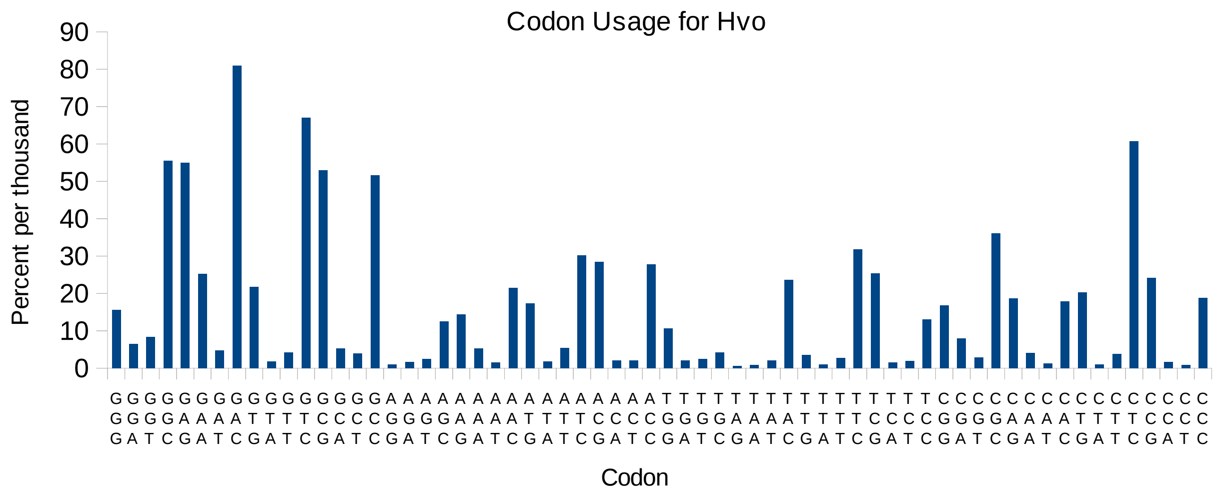

Figure 1. Codon usage graph for *Haloferax volcanii* (Hvo) depicting the absolute frequency of each codon using a percent per thousand score. This score was generated by tallying the occurrence of each codon in all open read frames (ORFs) of Hvo then dividing each tally by the total number of codons and multiplying by 1000.

While this method of codon optimization results in increased protein yields, recent evidence suggests that it alters protein function. We hypothesize that codon frequencies in ORFs model translational speed, are non-trivial to protein folding, and therefore play a major role in codon optimization. Because of this, we created our **F**requency **O**ptimized **C**odon **U**sage **S**trategy (FOCUS).

FOCUS provides the end user with a foundation to visualize the rate of protein synthesis as the nascent polypeptide exits the ribosome, granting potential insight to the underlying mechanics of protein folding. We plan to apply FOCUS accordingly to better understand the nature of misfolded proteins and the diseases associated, such as cancer and Alzheimers. FOCUS has the potential to identify detrimental single nucleotide polymorphisms that may be pivotal to the encoded protein's fold and serves as an extremely useful tool for protein engineering.

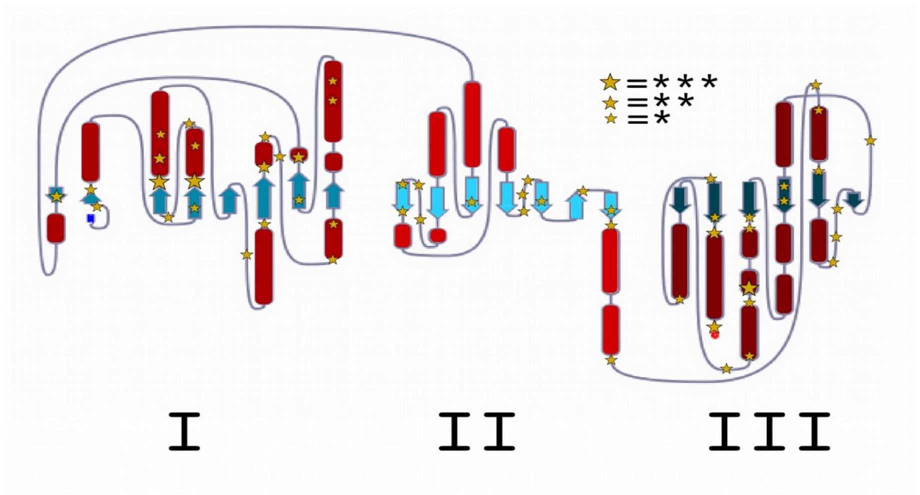

Figure 2: Secondary structure diagram of the Zymomonas mobilis pyruvate decarboxylase with rare codons indicated as gold stars. The three domains are labelled with Roman numerals starting from the N-terminus. Helicies are indicated by red and strands in blue.

We wanted our FOCUS program to codon optimize a protein sequence by matching the translational speed a much as possible. We wanted the genetic sequence to induce 'stalls' during translation by using a rare codon from the target organism's codon bias at specific locations. To achieve these stalls at the correct locations, we looked at the nucleotide sequence for the protein of interest from the native organism.

<u>Usage:</u>

- This program has been tested to work on Unix based operating systems (Mac OS, Linux)
    - When using IDLE, the script needs to be altered and the input file names need to be explicitly written. Variables that need to be hardcoded are commented in the FOCUS script.
- The input files need to be in the same directory as the FOCUS script.
    - The following program files **need** to be in the same directory for the FOCUS to work:
        - commandLine.py
        - fastaReader.py
        - FOCUS.py
        - parseTasser.py
        - GNUplot.py
- Codon bias table is generated using **CodonBiasGenerator**. However, you may use another tool to create the table, as long as it is a **GCG codon bias table**.
- Secondary structure prediction should be in I-TASSER format. Our team used the tool **PSSpred** to generate secondary structure predictions.

<u>Class FOCUS</u>

Attributes

- rnaCodonTable: Dictionary with RNA codons as keys and the single letter amino acid as the value.
- DnaCodonTable: Dicionary with DNA codons as keys and the single letter amino acid as the value.
- GCG: File name for the codon bias table given to the FOCUS object.
- CodonMap: Dictionary with the amino acid as the key and a list of lists as the value. The entries in the inner list are codon, relative frequency, and percent per thousand.

Functions

- OpenFile(seqFile): Opens a file and raises an error if it does not exist.
  - seqFile is the name of the text file
- CodonToAmino(codon): Converts a DNA codon to the encoding amino acid. The single amino acid letter is then returned
- MakeDict(): Fills CodonMap dictionary using the codon bias table.
- SortDict(dictionary) : For each amino acid in the CodonMap dictionary, the list of lists is sorted by the relative codon frequency, with the highest frequency being first.
- getScore(codon): Returns the relative codon score for a particular codon.
- printer(protien, values): Returns a string that has been formatted for printing to a file or terminal, where protein is a protein sequence string and values is a FOCUS score string for the same protein.
- SSprinter(protein, values, SSlist): Returns a string that has been formatted for printing to a file or terminal, much like printer() but with the addition of secondary structure prediction.

[BLOCK 4]

<u>Class parseTasser</u>

Attributes

- TASSER: filename for the secondary structure prediction in I-TASSER format.
- trans: Dictionary to translate the number for secondary structure to the corresponding character. ie. '1' corresponds to a coil, '2' corresponds to a helix, and '4' corresponds to a sheet.

Functions

- OpenFile(file): Opens the file and prints an error message if the file is not found.
- returnList(): Returns a list with the first entry being the secondary structure string and the second entry being the confidence for the secondary structure prediction.

<u>Class GNUplot</u>

Attributes

- proteinSeq: Protein sequence string
- freqSeq: FOCUS score string
- SSpred: List with a string for the Secondary structure prediction as the first entry.
- Thousand: List with entries correspoding to the percent per thousand. The first character in proteinSeq corresponds to the first entry in the Thousand list.
- reverseTrans: Dictionary to convert between Secondary Structure character to a number which can be used in GNUplot.

Functions

- makeTable(): Prints a table that can be used in GNUplot.